ENTERPRISE ARCHITECT

User Guide Series

Model Based Systems Engineering

Author: Sparx Systems

Date: 2021-07-19

Version: 15.2

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

Model Based Systems Engineering

Systems Engineering is, very broadly, the work of researching, designing and managing complex physical or electronic systems over their lifecycles. It focuses on the whole system and typically involves a number of sub-disciplines such as requirements, reliability, logistics, design, testing and maintenance; it considers not only the system itself but also processes, optimization and risk management, and requires sophisticated project management techniques.

In earlier decades a large but localized team might consider a very specific set of objects within a very specific and controlled environment, to be delivered to a small user base and maintained by perhaps an, again, localized team of experts who each might have responsibility for only a part of the system. Even for such a controlled and structured scenario, a huge volume of documentation was required to define the system requirements, the components, the engineering process, the standards applied and complied with, and the tests to be run on the system. Keeping this documentation cross-referenced, up to date and integrated was a major task.

Advancing into computing and basing Systems Engineering work on graphical models (Model Based Systems Engineering) provided huge benefits, allowing engineers to store and retrieve data from repositories, associate data with documentation also held in the repositories, and develop both master structures and variants from templates, all of which reduced the need to recreate and repeat work. The model initially represented the organization of the developing system, but grew to reflect the development process and the factors that supported and directed that process. As computing capabilities grew, and more specialized and sophisticated applications were made available, it became possible to represent the components of a system with increasingly varied and detailed model elements, and with increasingly varied and detailed relationships between them. Engineers could 'load' the model components and relationships with an array of properties, characteristics and parameters, which could be varied to reflect different scenarios. The standards that the system must apply or meet could be automatically enforced on the components as constraints, conditions and rules. More and more of the development process - such as testing - could be represented by element or model features, and more and more aspects of the process could be performed on the model by the application - such as automatic generation of code to make the system operational, and simulation of the system in action under various conditions.

These days, the Systems Engineer is likely to be a member of an interdisciplinary team that has to consider a wide range of factors in designing and modeling a system - a much broader, diverse and inexpert user base, a much broader maintenance base, how the system interacts with many other systems, how the system operates in many different and sometimes extreme environments, the impact the system has on the global environment - both within its operating framework and within its pre-use production and final disposal - the socio-economic environment controlling its acceptability and popularity, and how the system compares with its increasing range of competitors. To see how the work of the Systems Engineer has become vastly more complex one has only to think of a single development, such as the quantum leap from the relatively recent fixed-site landline telephone handset for making voice calls, to the modern mobile smartphone used as a camera, computer, cinema, music center, navigator, and audio, visual and text communicator.

Today, large projects and industries are being developed around systems and products for which the use cases are increasingly complex. Controlling this complexity grows further and further beyond the capacity of the engineer, increasing the level of risk to the product, the end user and the manufacturer. Examples of systems with dramatically increased risk include the manufacture of passenger air bags to be fitted to many different brands and types of car manufactured in different parts of the globe; or the requirements for the development of space probes intended to travel to the planets of the solar system and beyond.

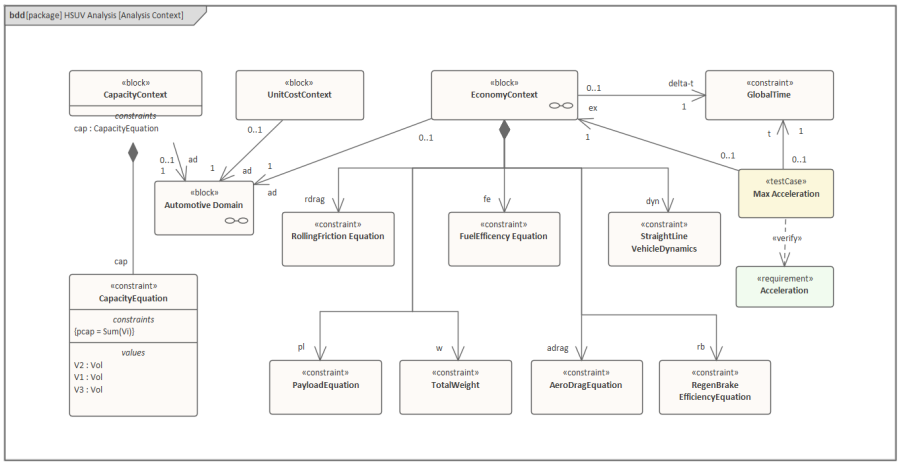
It is the advances in Systems Engineering tools and methodologies that have increased this complexity, whilst simultaneously providing the capability to manage and mitigate the associated risk, and reducing the difficulty and effort involved in managing and maintaining highly complex models.

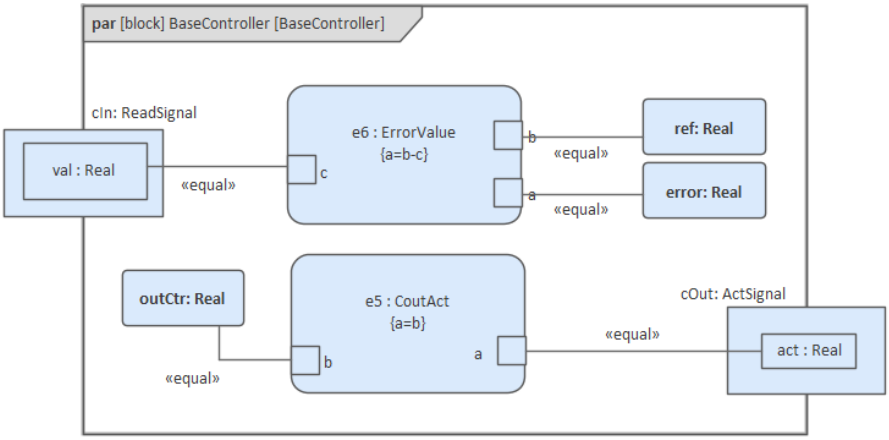
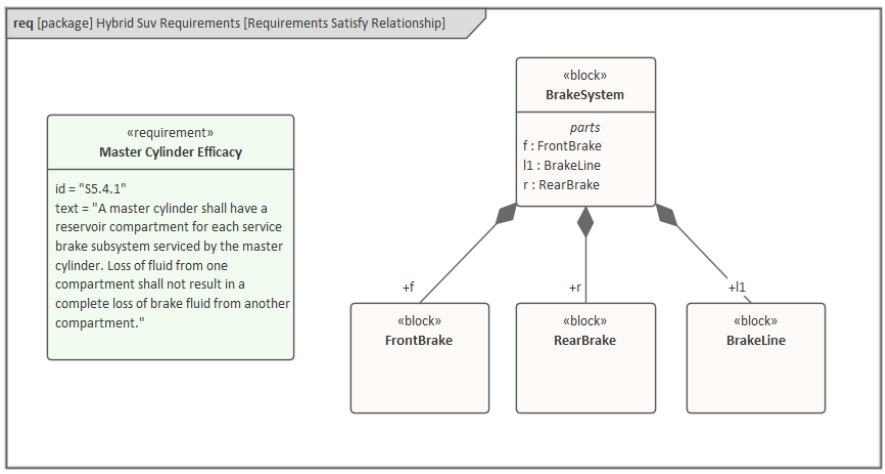
For additional information, see the *Representing Systems with Models* section of the 'SEBoK - Guide to the Systems Engineering Body of Knowledge' website.

Model-Based Systems Engineering in Enterprise Architect

Enterprise Architect provides a Model Based Systems Engineering platform that integrates many high-end features for Systems Engineers and model-based development, with these built-in features.

Feature	Description

SysML	Enterprise Architect is integrated with the Systems Modeling Language (SysML) versions 1.1, 1.2, 1.3, 1.4 and 1.5. For details, see the *Systems Modeling Language (SysML)* Help topic. Enterprise Architect provides a number of engineering model templates from which models of engineering structures and concepts can be developed. This is an image of a SysML 1.5 Block Definition diagram. It is part of the HSUV Model that can be found in the 'Systems Engineering' section of Enterprise Architect's Example Model. 
Conformance to Standards	As well as applying the standards defined by the OMG for UML and SysML, the Enterprise Architect Model Based Systems Engineering platform also complies with these international standards: • International Council of Systems Engineering (INCOSE) 2012 • Ontology Definition Metamodel (ODM) (OMG document *ptc/2013-12-03*, pub. February 2014) • Systems Modeling Language (SysML) (OMG document *formal/2017-05-01*) • Unified Profile for United States Department of Defense Architecture Framework (DoDAF) and United Kingdom Ministry of Defense Architecture Framework (MODAF) (UPDM) (OMG document *formal/2013-01-01*)
Executable Code Generation	You can quickly generate executable software code from your model elements, using Executable StateMachines. The code generated for an Executable StateMachine is based on its language property. This might be Java, C, C++, C# or JavaScript. Whichever language it is, Enterprise Architect generates the appropriate code, which is immediately ready to build and run. There are no manual interventions necessary before you run it. For more information, see the *Code Generation for Executable StateMachines* Help topic.
Model to Code Transformations for HDLs	You can not only generate executable software code, but you can generate Hardware Description Languages and Ada from your model elements, for the chips and circuits in system hardware components. For more information, see the *StateMachine Modeling for HDLs* Help topic.
Parametric Model Simulation	Enterprise Architect provides facilities to create Parametric diagrams using the Parametric Diagram Modeling Assistant, and to perform Parametric Model Simulation through OpenModelica. Being able to simulate a system through the model is a huge advantage where live-testing would be dangerous (defense systems) or prohibitively expensive (space probes). This image shows an Internal Block diagram used in a Parametric Model Simulation. The diagram is part of the 'Two Tanks' example that can be found in the 'Systems Engineering > Modelica Examples' section of Enterprise Architect's

	Example Model.  For further information, see the *Parametric Diagrams*, *Parametric Diagram Modeling Assistant* and *Parametric Simulation Using OpenModelica* Help topics.
System-of-Systems Modeling	In addition to developing system models, you can also design 'system-of-system' models, or system architectures, using the Unified Profile for DoDAF and MODAF (UPDM) or the Unified Architecture Framework (UAF); these are both accessible through the Systems Engineering Perspective with SysML.
Requirements Management	Enterprise Architect has an extensive suite of Requirements Management tools that can be applied to System Engineering, dove-tailed to the SysML Requirements modeling facility. See the *SysML Requirements Model* and *Requirement Models* Help topics. This image shows an example of a SysML Requirements diagram. 
Project Management	Enterprise Architect has extensive Project Management and team support facilities to help you organize, support and manage both the Systems Engineering model content and the staff working on the project. Amongst other things, you can apply user security, organize and monitor resources, schedule tasks, apply Version Control and enable a range of discussions from simple messaging through informal topic discussion threads to formal reviews. For more information, see the *Project Management* and *The Modeling Team* Help sections.